Unsupervised Learning

22c:145 Artificial Intelligence

The University of Iowa

Supervised learning vs. unsupervised learning

- Supervised learning: discover patterns in the data that relate data attributes with a target (class) attribute.
  - These patterns are then utilized to predict the values of the target attribute in future data instances.
- Unsupervised learning: The data have no target attribute.
  - We want to explore the data to find some intrinsic structures in them.

2

What is Clustering?

Also called *unsupervised learning*, sometimes called *classification* by statisticians and *sorting* by psychologists and *segmentation* by people in marketing

- Organizing data into classes such that there is
  - high intra-class similarity
  - low inter-class similarity
- Finding the class labels and the number of classes directly from the data (in contrast to classification).
- More informally, finding natural groupings among objects.

What is clustering for?

- Let us see some real-life examples
- Example 1: groups people of similar sizes together to make "small", "medium" and "large" T-Shirts.
  - Tailor-made for each person: too expensive
  - One-size-fits-all: does not fit all.
- Example 2: In marketing, segment customers according to their similarities
  - To do targeted marketing.

4

What is clustering for? (cont…)

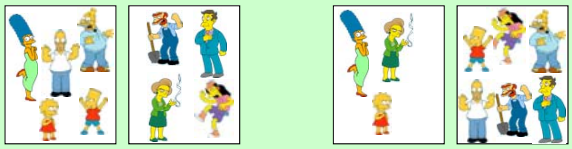
- Example 3: Given a collection of text documents, we want to organize them according to their content similarities,
  - To produce a topic hierarchy
- In fact, clustering is one of the most utilized data mining techniques.
  - It has a long history, and used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc.
  - In recent years, due to the rapid increase of online documents, text clustering becomes important.

5

What is a natural grouping among these objects?

What is a natural grouping among these objects?

Clustering is subjective



Simpson's Family | School Employees | Females | Males

What is Similarity?

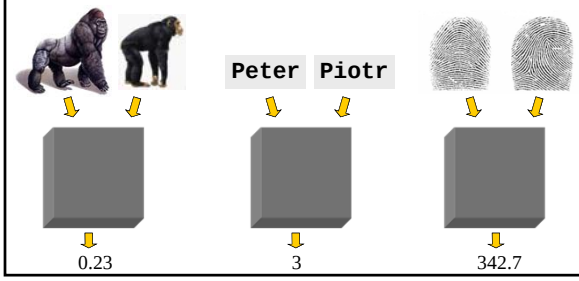
The quality or state of being similar; likeness; resemblance; as, a similarity of features.
Webster's Dictionary

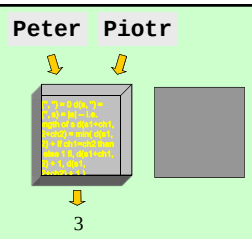
Similarity is hard to define, but…
"*We know it when we see it*"

The real meaning of similarity is a philosophical question. We will take a more pragmatic approach.

Defining Distance Measures

Definition: Let O_1 and O_2 be two objects from the universe of possible objects. The distance (dissimilarity) between O_1 and O_2 is a real number denoted by $D(O_1,O_2)$





When we peek inside one of these black boxes, we see some function on two variables. These functions might very simple or very complex.
In either case it is natural to ask, what properties should these functions have?

What properties should a distance measure have?

- $D(A,B) = D(B,A)$ *Symmetry*
- $D(A,A) = 0$ *Constancy of Self-Similarity*
- $D(A,B) = 0$ iff A= B *Positivity (Separation)*
- $D(A,B) \leq D(A,C) + D(B,C)$ *Triangular Inequality*

Intuitions behind desirable distance measure properties

$D(A,B) = D(B,A)$ *Symmetry*
Otherwise you could claim "Alex looks like Bob, but Bob looks nothing like Alex."

$D(A,A) = 0$ *Constancy of Self-Similarity*
Otherwise you could claim "Alex looks more like Bob, than Bob does."

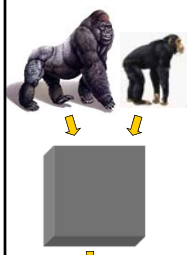
$D(A,B) = 0$ iff A=B *Positivity (Separation)*
Otherwise there are objects in your world that are different, but you cannot tell apart.

$D(A,B) \leq D(A,C) + D(B,C)$ *Triangular Inequality*
Otherwise you could claim "Alex is very like Bob, and Alex is very like Carl, but Bob is very unlike Carl."

Desirable Properties of a Clustering Algorithm

- Scalability (in terms of both time and space)
- Ability to deal with different data types
- Minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers
- Insensitive to order of input records
- Incorporation of user-specified constraints
- Interpretability and usability

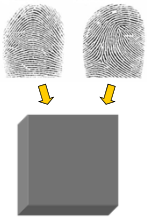
How do we measure similarity?



0.23

Peter Piotr

3



342.7

A generic technique for measuring similarity

To measure the similarity between two objects, transform one of the objects into the other, and measure how much effort it took. The measure of effort becomes the distance measure.

The distance between Patty and Selma.

```
Change dress color,    1 point
Change earring shape,  1 point
Change hair part,      1 point
```

D(Patty,Selma) = **3**

The distance between Marge and Selma.

```
Change dress color,    1 point
Add earrings,          1 point
Decrease height,       1 point
Take up smoking,       1 point
Lose weight,           1 point
```

D(Marge,Selma) = **5**

This is called the "edit distance" or the "transformation distance"

Edit Distance Example

It is possible to transform any string *Q* into string *C*, using only *Substitution*, *Insertion* and *Deletion*. Assume that each of these operators has a cost associated with it.

The similarity between two strings can be defined as the cost of the cheapest transformation from *Q* to *C*.

Note that for now we have ignored the issue of how we can find this cheapest transformation.

How similar are the names "Peter" and "Piotr"?

Assume the following cost function

Substitution	1 Unit
Insertion	1 Unit
Deletion	1 Unit

D(**Peter**,**Piotr**) is 3

Peter
↓ Substitution (i for e)
Piter
↓ Insertion (o)
Pioter
↓ Deletion (e)
Piotr

Partitional Clustering

- Nonhierarchical, each instance is placed in exactly one of K nonoverlapping clusters.
- Since only one set of clusters is output, the user normally has to input the desired number of clusters K.

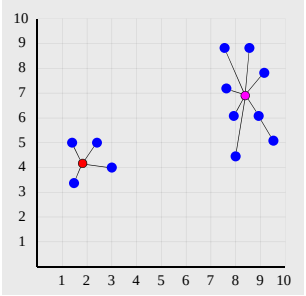
Minimize Squared Error

Distance of a point *i* in cluster *k* to the center of cluster *k*

$$se_{K_i} = \sum_{j=1}^{m} \|t_{ij} - C_k\|^2$$

$$se_K = \sum_{j=1}^{k} se_{K_j}$$

Objective Function



K-means clustering

- K-means is a partitional clustering algorithm
- Let the set of data points (or instances) *D* be

  $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$,

  where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a vector in a real-valued space $X \subseteq R^r$, and *r* is the number of attributes (dimensions) in the data.
- The *k*-means algorithm partitions the given data into *k* clusters.
  - Each cluster has a cluster **center**, called **centroid**.
  - *k* is specified by the user

Algorithm *k-means*

1. Decide on a value for *k*.

2. Initialize the *k* cluster centers (randomly, if necessary).

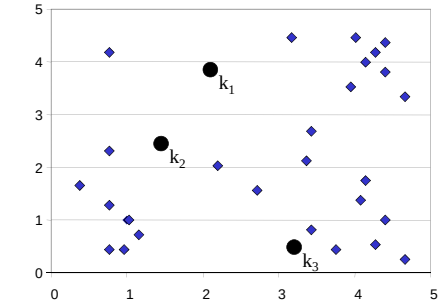
3. Decide the class memberships of the *N* objects by assigning them to the nearest cluster center.

4. Re-estimate the *k* cluster centers, by assuming the memberships found above are correct.

5. If none of the *N* objects changed membership in the last iteration, exit. Otherwise goto 3.

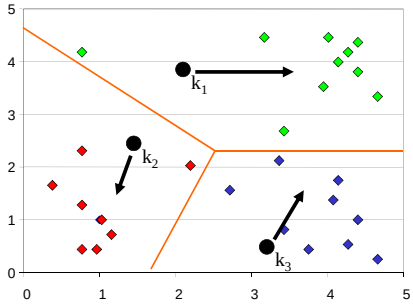
K-means Clustering: Step 1

Algorithm: k-means, Distance Metric: Euclidean Distance



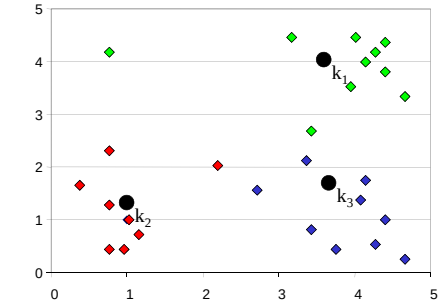
K-means Clustering: Step 2

Algorithm: k-means, Distance Metric: Euclidean Distance



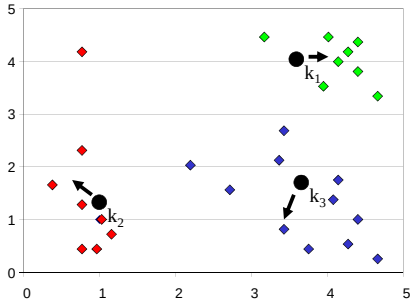
K-means Clustering: Step 3

Algorithm: k-means, Distance Metric: Euclidean Distance



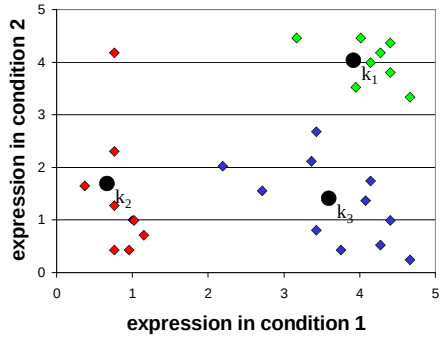
K-means Clustering: Step 4

Algorithm: k-means, Distance Metric: Euclidean Distance



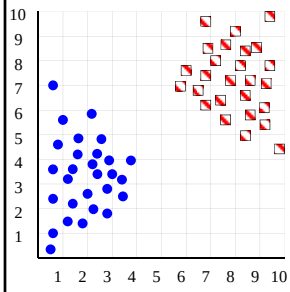
K-means Clustering: Step 5

Algorithm: k-means, Distance Metric: Euclidean Distance



How can we tell the *right* number of clusters?

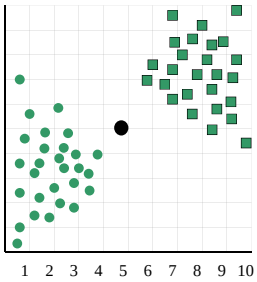
In general, this is an unsolved problem. However there are many approximate methods. In the next few slides we will see an example.



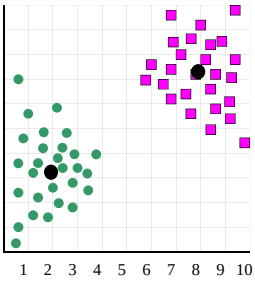
For our example, we will use the dataset on the left.

However, in this case we are imagining that we do NOT know the class labels. We are only clustering on the X and Y axis values.

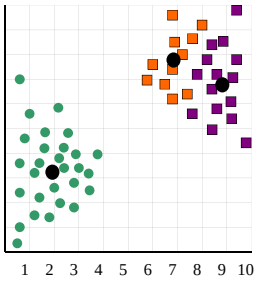
When k = 1, the objective function is 873.0



When k = 2, the objective function is 173.1

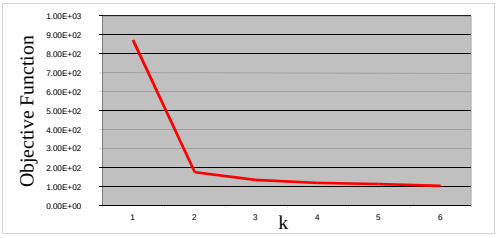


When k = 3, the objective function is 133.6



We can plot the objective function values for k equals 1 to 6…

The abrupt change at k = 2, is highly suggestive of two clusters in the data. This technique for determining the number of clusters is known as "knee finding" or "elbow finding".



Note that the results are not always as clear cut as in this toy example

"elbow finding" implementation

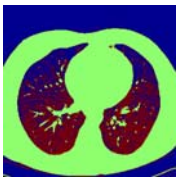
$$se_{K_i} = \sum_{j=1}^{m} \|t_{ij} - C_k\|^2$$

$$se_K = \sum_{j=1}^{k} se_{K_j}$$

New objective function: Find *k* such that
$se_K + ck$
is minimized, where *c* is a constant.

Image Segmentation Results

An image (*I*)



Three-cluster image (*J*) on gray values of *I*

Note that *K*-means result is "noisy"

Strengths of k-means

- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: $O(tkn)$,
    where n is the number of data points,
    k is the number of clusters, and
    t is the number of iterations.
  - Since both k and t are small, k-means is considered a linear algorithm.
- K-means is the most popular clustering algorithm.
- Note that: it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

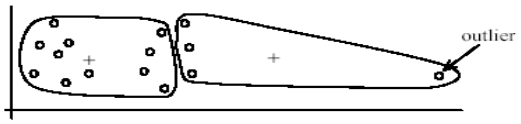
32

Weaknesses of k-means

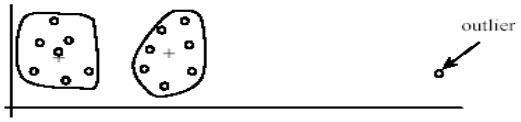
- The algorithm is only applicable if the mean is defined.
  - For categorical data, *k*-mode - the centroid is represented by most frequent values.
- The user needs to specify *k*.
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.

33

Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



(B): Ideal clusters

Weaknesses of k-means: To deal with outliers

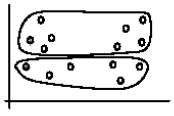
- One method is to remove some data points in the clustering process that are much further away from the centroids than other data points.
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
- Another method is to perform random sampling. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

35

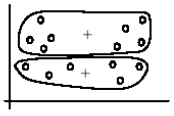
Weaknesses of k-means (cont …)

- The algorithm is sensitive to initial seeds.

(A). Random selection of seeds (centroids)



(B). Iteration 1

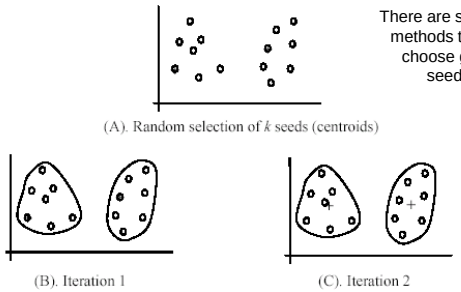


(C). Iteration 2

36

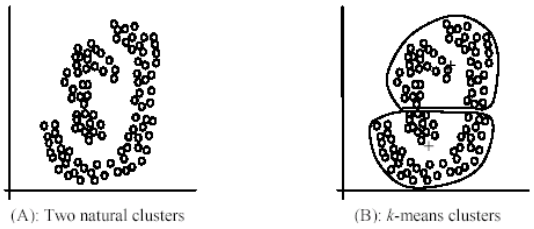
Weaknesses of k-means (cont …)

- If we use different seeds: good results



Weaknesses of k-means (cont …)

- The *k*-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).



K-means summary

- Despite weaknesses, *k*-means is still the most popular algorithm due to its simplicity, efficiency and
  - other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general
  - although they may be more suitable for some specific types of data or applications.
- Comparing different clustering algorithms is a difficult task. No one knows the correct clusters!

K-Means Summary (cont)

- Strength
  - *Relatively efficient training*: *O*(*tknm*), where *n* is # objects, *m* is size of an object, *k* is # of clusters, and *t* is # of iterations. Normally, *k*, *t* << *n*.
  - *Efficient decision*: O(*km*)
  - Often terminates at a *local optimum*. The *global optimum* may be found using techniques exhaustive search.
- Weakness
  - Applicable only when *mean* is defined. What about categorical data?
  - Need to specify *k*, the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

Vector Quantization

Voronoi Region

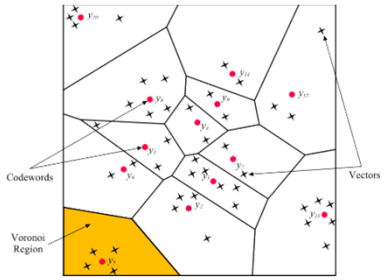
- Blocks:
  - A sequence of audio signals.
  - A block of image pixels.

  Formally called a **vector.** Example: (0.2, 0.3, 0.5, 0.1)
- A vector quantizer maps *k-dimensional* vectors of the vector space R^k into a finite set of vectors $Y = \{y_i: i = 1, 2, \ldots, K\}$.
- Each vector y_i *is* called a code vector or a *codeword*. and the set of all the codewords is called a *codebook*.
- Associated with each codeword, y_i, is a nearest neighbor region called *Voronoi* region, and it is defined by:

$$V_i = \left\{x \in R^k : \|x - y_i\| \le \|x - y_j\|, \forall j \ne i\right\}$$

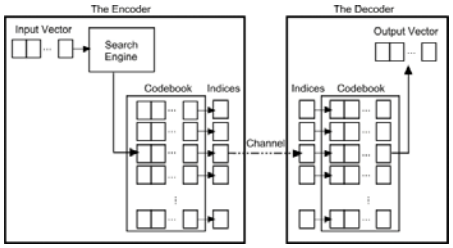
- The set of Voronoi regions partition the entire space R^k .

Two Dimensional Voronoi Diagram



Codewords in 2-dimensional space. Input vectors are marked with an x, codewords are marked with red circles, and the Voronoi regions are separated with boundary lines.

The Schematic of a Vector Quantizer for Signal Compression



Vector Quantization Algorithm (identical to k-means Algorithm)

1. **Determine the number of codewords, *K*, or the size of the codebook**.
2. **Select *K* codewords at random, and let that be the initial codebook**. The initial codewords can be randomly chosen from the set of input vectors.
3. **Using the scaled Euclidian distance measure clusterize the vectors around each codeword**. This is done by taking each input vector and finding the scaled Euclidian distance between it and each codeword. The input vector belongs to the cluster of the codeword that yields the minimum distance.

VQ Algorithm (contd.)

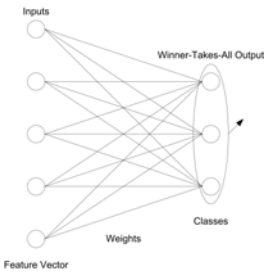
4. **Compute the new set of codewords**. This is done by obtaining the average of each cluster. Add the component of each vector and divide by the number of vectors in the cluster.

$$y_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij}$$

where *i* is the component of each vector (x, y, z, ... directions), *m* is the number of vectors in the cluster.

5. **Repeat steps 2 and 3 until the either the codewords don't change or the change in the codewords is small**.

Regard VQ as Neural Network



Weights define the center of each cluster. Can be adjusted by (Eq 9.3):

$w_{M,i} \mathrel{+}= alpha(x_i - w_{M,i})$

(Eq 9.2) is the same as (Eq 8.13) for PNN, not Euclidian distance.

Adaptive Resonance Theory (ART)

Objectives

- There is no guarantee that, as **more** inputs are applied to a neural network, the weight matrix will eventually **converge**. The same problem exists for k-means or vector quantitation.
- Present a modified type of competitive learning, called **Adaptive Resonance Theory** (ART), which is a collection of models for unsupervised learning and designed to overcome the problem of learning stability.

Theory & Examples

- A key problem of k-means algorithm and the vector quantitation is that they do NOT always form stable clusters (or categories).
- The learning instability occurs because of the network's **adaptability** (or **plasticity**), which causes *prior learning* to be **eroded** by more *recent learning*.

Stability / Plasticity

- How can a system be receptive to significant new patterns and yet remain stable in response to irrelevant patterns?
- Grossberg and Carpenter developed the ART to address the stability/plasticity dilemma.
  - The ART networks are similar to Vector Quantitation, or k-means algorithm.

Key Innovation

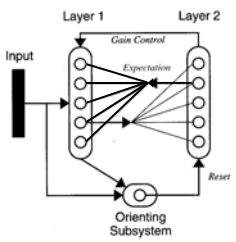
The key innovation of ART is the use of "**expectations**."

- As each **input** is presented to the network, it is compared with the **cluster** that is most closely **matches** (the expectation).
- If the **match** between the cluster and the input vector is ***NOT*** adequate, a **new cluster** is created. In this way, *previous learned memories (old clusters) are **not** eroded by new learning*.

Algorithm ART

1. Initialize: let the first object be the only cluster center.
2. For each object, choose the nearest cluster center.
3. If the distance to this cluster center is acceptable, add this object to the cluster and adjust the cluster center.
4. If the distance is too big, create a new cluster for this object.
5. Repeat 2-4 until no new clusters are created and no objects change clusters.

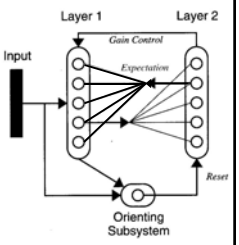
Neural Network Model



Basic ART architecture

ART Network

- The Layer1-Layer2 connections perform a **clustering** (or **categorization**) operation. When an input pattern is presented, the normalized distance between the input vector and the nodes in Layer 2 are computed.
- A **competition** is performed at Layer 2 to determine which node is closest to the input vector. If the distance is acceptable, the weights are updated so that node is then moved toward the input vector.
- If no acceptable nodes are present, the input vector will become a new node of Layer 2.



ART Types

- ART-1
  - Binary input vectors
  - Unsupervised NN that can be complemented with external changes
- ART-2
  - Real-valued input vectors

Normalized Distance for ART-1

1. Compute $\mathbf{a} = \mathbf{p} \cap \mathbf{w}_j$ $\mathbf{w}_j$ is the cluster center; $\mathbf{p}$ input
2. $\begin{cases} if \left(\|\mathbf{a}\|^2 / \|\mathbf{p}\|^2 \right) \geq \rho & \text{update cluster center as } \mathbf{a}. \\ else & \text{create a new cluster center.} \end{cases}$

where $\|\mathbf{x}\|$ is # of 1's in $\mathbf{x}$. ρ is vigilance factor.

We may use $x \cap y = \min(x, y)$ so real numbers are accepted.
And we update $\mathbf{w}_j$ as $\mathbf{w}_j := (1 - \gamma)\mathbf{w}_j + \gamma \mathbf{p}$.

An Example of Associative Networks: Hopfield Network

- John Hopfield (1982)
  - Associative Memory via artificial neural networks
  - Solution for optimization problems
  - Statistical mechanics

Neurons in Hopfield Network

- The neurons are binary units
  - They are either active (1) or passive (-1)
  - Alternatively 1 or 0
- The network contains *N* neurons
- The state of the network is described as a vector of 1s and -1s:

$$U = (u_1, u_2, ..., u_N) = (-1, 1, -1, 1, ..., -1, -1, 1)$$

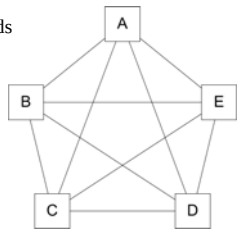
- There are input states and output states.

Architecture of Hopfield Network

- The network is fully interconnected
  - All the neurons are connected to each other
  - The connections are bidirectional and symmetric

$$W_{i,j} = W_{j,i}$$

  - The setting of weights depends on the application

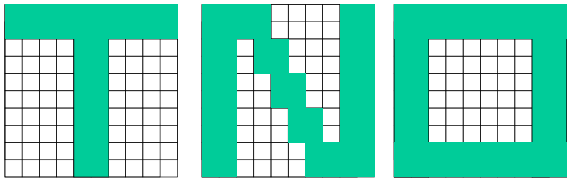


Hopfield network as a model for associative memory

- Associative memory
  - Associates different features with each other
    - Karen ←→ green
    - George ←→ red
    - Paul ←→ blue
  - Recall with partial cues

Neural Network Model of associative memory

- Neurons are arranged like a grid: green square means 1; blank square means -1



Setting the weights

- Each pattern can be denoted by a vector of -1s and 1s: $E_p = (-1,1,-1,1,...,-1,-1,1) = (e_1^p, e_2^p, e_3^p, ...e_N^p)$
- If the number of patterns is *m* then:
- $w_{i,j} = 0$; $w_{i,j} += \sum_{p=1}^{m} e_i^p e^p{}_j$ for i != j.
- We may use the shorthand $W = \sum_{p=1}^{m} E_p{}^T E_p$
- Hebbian Learning:
  - The neurons that fire together, wire together

Updating States

- There are many ways to update states of Hopfield network. And updating may be continued until a stable state (i.e., X = sgn(XW)) is reached.
  - For a given input state *X*, each neuron receives a weighted sum of the input state from the weights of other neurons:

$$h_j = \sum_{\substack{i=1 \\ i \neq j}}^{N} x_i . w_{j,i}$$

  - If the input h_j is positive the new state of the neuron will be 1, otherwise 0, i.e., $y_j = \text{sgn}(h_j)$.

$$y_j = \begin{cases} 1 & \text{if } h_j \geq 0 \\ -1 & \text{if } h_j < 0 \end{cases} \quad \text{or } Y = \text{sgn}(XW)$$

Limitations of Hofield associative memory

- The recalled pattern is sometimes not necessarily the most similar pattern to the input
- Some patterns will be recalled more than others
- Spurious states: non-original patterns
- Capacity: *0.15 N* of stable states